Overview of the approach used

Our main approach for the project is to create our pipelines which are where we send data from our source to a target as data happens in real-time as streams of events(data). This approach of creating and using a pipeline will help us get more accurate and more up to data in the target system. Also, you only have to wait until data is propagated through the pipeline and get to react to data as it changes. Additionally enables you to get more economical use of your computing and storage resources.

In regards to Kafka streaming data, the pipeline is means of ingesting the data from some source system into Kafka as the data is created. And then streaming that data from Kafka to one or more target systems. Because Kafka is a distributed system it is quite scalable and resilient and serves as a distributed log(data storage system) here. Some of the benefits of decoupling the source system from the target system using Kafka in our pipeline are:

- Kafka can store our data as long as we want: so if the target system goes down there will not be any long-term permanent impact on our pipeline. Also if for any reason the source system goes offline there won't be any issues to our pipeline and the target system may not even know it's actually of line, there just might be less traffic or no traffic due to the source system being offline. It will resumes from the point it went down out with no problem when it back up.
- Pipelines created with Kafka can evolve smoothly a piece at a time: since the data stored by Kafka can be sent to multiple targets independently.

In our pipeline, we will have a :

- Text selector(producer) that will input text corpus from S3 and output a single selected text to Kafka topic1.
- the Kafka topic1 has single selected text input and outputs the text upon request
- On the Web application, the consumer will view the selected text
- On the Web application, the producer will create an audio file from the user and that goes to a Kafka topic2
- Kafka topic2 will have an audio file input and output it upon a request

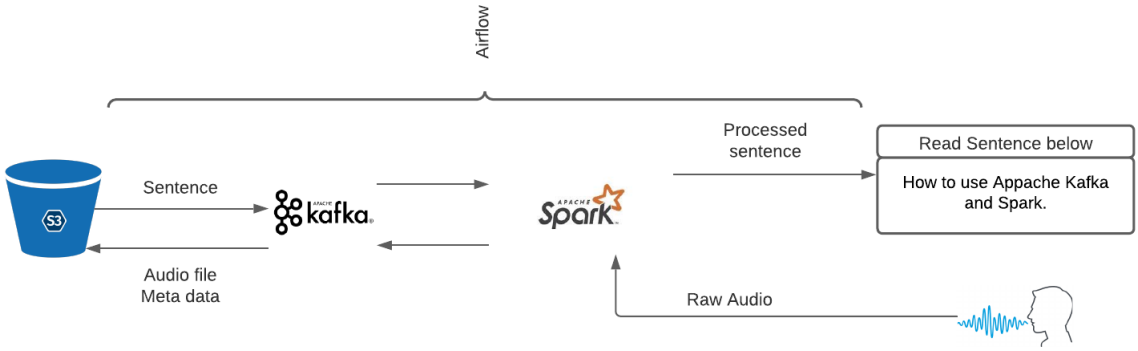

diagram of our pipeline overview

Why use Apache Kafka?

Is used as a transportation mechanism to move data real fast in scale. Apache Kafka is a distributed event store and stream-processing platform. It addresses the complexity and increased load when having many source systems and target systems exchanging data. Kafka has high-performance meaning latency of less than 10 ms and hight-thoruhput as well.

In Kafka, we have topics that are a particular stream of data similar to a table in the database and can have many of them. These topics are split into partitions which are part of topics and get incremental id called offsets. Once data is written its immutable and data is assigned randomly to a partition unless a key is provided. Kafka cluster is composed of many brokers(servers) each having a unique id. Once connected to one broker you can have access to other brokers in the cluster. Topics should have a replication factor of at least 2 so that if one server(broker) is down the others can serve the whole data. Additionally, at any time only one leader for a given partition which means only that leader usually indicated by a start can receive and serve data. The rest of the brokers are in-sync replica(ISR). The brokers will be managed by Zookeepers.

When it comes to producers they are to write data on topics. Producers can choose acknowledgments

Ack = 0: producers won't wait for any acknowledgment (possibility of data loss)

Ack =1: producers will wait for leader acknowledgment(limited data loss)

Act = All: producers will wait for both leader and replicas acknowledgment